2009

Securing location discovery in wireless sensor networks

Wisam F. Kadhim

Securing Location Discovery in Wireless Sensor Networks

by

Wisam F. Kadhim

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Supervised by

Dr. Minseok Kwon
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY

August 2009

Approved By:

Dr. Minseok Kwon
Advisor – R.I.T. Dept. of Computer Science

Dr. Zack Butler
Reader – R.I.T. Dept. of Computer Science

Dr. Hans-Peter Bischof
Observer – R.I.T. Dept. of Computer Science

Thesis Release Permission Form

Rochester Institute of Technology
Golisano College of Computing and Information Sciences

Title: Securing Location Discovery in Wireless Sensor Networks

I, Wisam F. Kadhim, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

_____
Wisam F. Kadhim

_____
Date

Dedication

To my family…
May God protect you and cast his blessings upon you…

Acknowledgements

So many people have influenced who I am today and have helped guide me to this point. The following is only a small subset of those who deserve my heartfelt thanks. My thanks to Professor Kwon for his unwavering guidance during my work. Also, to Professor Bischof for all the extraordinary support he has given me. To Professor Steele and Professor Schreiner, for encouraging new ways of thinking and indirectly pushing me to explore my artistic side, and to Professor Butler for reviewing this work. I also thank Jaime Burns and Ara Murad for graciously editing several drafts and pushing me towards the finish line. Thank you all so much.

Abstract

Providing security for wireless sensor networks in hostile environments has a significant importance. Resilience against malicious attacks during the process of location discovery has an increasing need. There are many applications that rely on sensor nodes' locations to be accurate in order to function correctly. The need to provide secure, attack resistant location discovery schemes has become a challenging research topic. In this thesis, location discovery techniques are discussed and the security threats and attacks are explained. I also present current secure location discovery schemes which are developed for range-based location discovery.

The thesis goal is to develop a secure range-free location discovery scheme. This is accomplished by enhancing the voting-based scheme developed in [8, 9] to be used as the bases for developing a secure range-free location discovery scheme. Both the enhancement voting-based and the secure range-free schemes are implemented on Sun SPOT wireless sensors and subjected to various levels of location discovery attacks and tested under different sensor network scales using a simulation program developed for testing purposes.

Table of Contents

List of Figures

List of Tables

Glossary

AoA	Angle of Arrival
APIT	Area based Point In Triangulation
BARMMSE	Brute-force Attack Resistant Minimum Mean Square Error
CLDC	Connected Limited Device Configuration
CoG	Center of Gravity
EARMMSE	Enhanced greedy Attack Resistant Minimum Mean Square Error
GARMMSE	Greedy Attack Resistant Minimum Mean Square Error
GPS	Global Positioning System
GUI	Graphical User Interface
LED	Light Emitting Diode
MIDP	Mobile Information Device Profile
MMSE	Minimum Mean Square Error
OS	Operating System
PIT	Point in Triangulation
RADAR	An In-Building RF-based User Location and Tracking System
RGB	Red/Green/Blue
ROPE	Rubust Position Estimation
RSSI	Received Signal Strength Indicator
SeRLoc	Secure Range independent Localization
SPINE	Secure Positioning In sensor Networks
SPOT	Small Programmable Object Technology
TDoA	Time Difference of Arrival
ToA	Time of Arrival
VM	Virtual Machine
WSN	Wireless Sensor Networks

Chapter 1

INTRODUCTION

Wireless Sensor Networks (WSN) are networks made of small, battery-powered, memory-constrained devices called sensor nodes, which have the capability of wireless communication over a restricted area. Due to memory and power constraints, they need to be well-arranged to build a fully functional network. Wireless sensors' locations are vital to many sensor network applications such as environmental monitoring, military applications, and many other applications which require sensors' location information to fulfill their tasks. There are also several fundamental techniques [8] developed for wireless sensor networks which require wireless sensor nodes' locations, for instance geographic routing protocols where sensor nodes make routing decisions based on their own location as well as their neighbors' locations.

Despite recent advances, location discovery for wireless sensor networks in hostile environments has been typically overlooked. Most of the existing location discovery protocols are vulnerable in the presence of malicious attacks. An attacker may provide incorrect location references by replaying the beacon packets intercepted in different locations. Furthermore, an attacker may compromise a beacon node and distribute malicious location references by lying about the beacon node’s location or manipulating the beacon signals. In either of these cases, non-beacon nodes will determine their locations incorrectly. The security of location discovery can certainly be enhanced by authentication. However, authentication does not guarantee the security of location discovery. An attacker may forge beacon packets with keys learned through compromised nodes, or replay beacon signals intercepted in different locations [8].

Several attack-resistant location estimation techniques were developed to tolerate the malicious attacks against range-based location discovery in wireless sensor networks. This thesis focuses on the voting-based location estimation technique developed in [8, 9], where the deployment field is quantized into a grid of cells and has each location reference vote on the cells in which the node may reside with iterative refinement of the voting results so that it can be executed in resource constrained sensor nodes.

The thesis work provides enhancement over the voting-based location discovery estimation to simplify it further and make it adaptable for use in range-free schemes. The enhanced voting-based technique would be applied to develop a secure range-free

location discovery scheme. This would provide a less complicated, cost-effective, and more applicable solution for wireless sensor networks.

1.1 Classification of Localization Techniques

1.1.1 Direct approaches

This is also known as *absolute localization*. The direct approach itself can be classified into two types: *Manual configuration* and *GPS-based localization*. The manual configuration method is very cumbersome and expensive. It is neither practical nor scalable for large scale WSNs and in particular, does not adapt well for WSNs with node mobility. On the other hand, in the GPS-based localization method, each sensor is equipped with a GPS receiver. This method adapts well for WSNs with node mobility.

However, there is a downside to this method. It is not economically feasible to equip each sensor with a GPS receiver since WSNs are deployed with hundreds of thousands of sensors. This increases the size of each sensor, rendering them unfit for pervasive environments. Also, the GPS receivers only work well outdoors on earth and have line-of-sight requirement constraints. Such WSNs cannot be used for underwater applications like habitat monitoring, water pollution level monitoring, and tsunami monitoring [1].

1.1.2 Indirect approaches

The indirect approach of localization is also known as *relative localization* since nodes position themselves relative to other nodes in their vicinity. The indirect approaches of localization were introduced to overcome some of the drawbacks of the GPS-based direct localization techniques while retaining some of their advantages, like accuracy of localization. In this approach, a small subset of nodes in the network, called the *beacon nodes*, are either equipped with GPS receivers to compute their location or are manually configured with their location. These beacon nodes then send beams of signals providing their location to all sensor nodes in their vicinity that do not have a GPS receiver. Using the transmitted signal containing the location information, sensor nodes compute their location. This approach effectively reduces the overhead introduced by the GPS-based method [1].

However, since the beacon nodes are also operating in the same hostile environment as the sensor nodes, they too are vulnerable to various threats, including physical capture by adversaries. This introduces new security threats concerning the honesty of the beacon nodes in providing location information since they could have been tampered with by the adversary and misbehaves by providing incorrect location information [1].

Within the indirect approach, the localization process can be classified into the following two categories:

Range-based:

In range-based localization, the location of a node is computed relative to other nodes in its vicinity. Range-based localization depends on the assumption that the absolute distance between a sender and a receiver can be estimated by one or more features of the communication signal from the sender to the receiver. The accuracy of such estimation, however, is subject to the transmission medium and surrounding environment. Range based techniques usually rely on complex hardware which is not feasible for WSNs since sensor nodes are highly resource-constrained and have to be produced at throw-away prices as they are deployed in large numbers. The features of the communication signal that are frequently used in literature for range-based localization are as follows:

- Angle of Arrival (AoA): Range information is obtained by estimating and mapping relative angles between neighbors.
- Received Signal Strength Indicator (RSSI): Use a theoretical or empirical model to translate signal strength into distance. RADAR [1, 11] is one of the first to make use of RSSI.
- Time of Arrival (ToA): To obtain range information using ToA, the signal propagation time from source to destination is measured. A GPS is the most basic example that uses ToA. To use ToA for range estimation, a system needs to be synchronous, which necessitates the use of expensive hardware for precise clock synchronization with the satellite.
- Time Difference of Arrival (TDoA): To obtain the range information using TDoA, an ultrasound is used to estimate the distance between the node and the source. Like ToA, TDoA necessitates the use of special hardware, rendering it too expensive for WSNs.

Range-free:

Range-free localization never tries to estimate the absolute point to point distance based on received signal strength or other features of the received communication signal like time, angle, etc. This greatly simplifies the design of hardware, making range-free methods very appealing and a cost-effective alternative for localization in WSNs. Amorphous localization [12], Centroid localization [10], APIT [18], DV-Hop localization [3], SeRLoc [5] are some examples of range-free localization techniques [1].

1.2 Security Threats Associated with Location Discovery

In hostile environments, an adversary can compromise sensor nodes location discovery via injecting misleading location references. The attacker can either be an insider or an outsider. As an insider, the attacker has access to all of the cryptographic keying material held by a node. This is potentially dangerous since the attacker can now claim to be a legitimate part of the network. Authentication or verification via password and other mechanisms give up under this attack model. On the other hand, in the outsider attack model, the attacker is outside the network and has no information about cryptographic keys and passwords necessary for authentication. The attacker can only capture a node but cannot extract the sensitive information. This model is comparatively less detrimental, but harmful nonetheless. So, for localization process to be secured it has to be robust in its defense against both outsider and insider attacks [1].

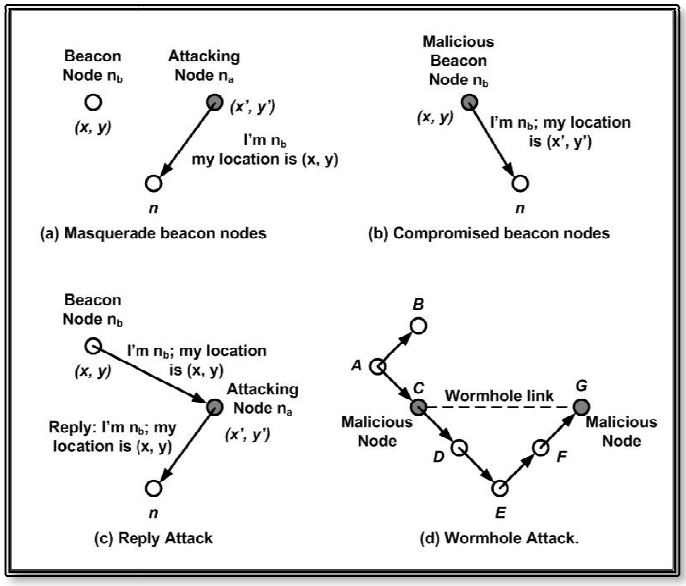

Figure 1.1: *Attack patterns against location discovery schemes [1, 8].*

Some attacks that have been discussed for nearly a decade in literature that are the most common against localization schemes are as follows:

- **Masquerading and Compromising Beacon Nodes:** In masquerading, the attacker would impersonate a beacon node and send misleading information about its location to divert the other benign nodes from discovering their accurate location, as shown in Figure 1.1(a). An attacker can also compromise beacon nodes through physical capture, and use the compromised beacon nodes to inject

misleading location references that would affect location discovery for other benign nodes, as shown in Figure 1.1(b) [8].

- **Replay Attack:** A replay attack is the easiest and most commonly used by attackers. Specifically, when an attacker's capability is limited, i.e., the attacker cannot compromise more than 1 node; this is the most preferred attack. In a replay attack, the attacker merely jams the transmission between a sender and a receiver and later replays the same message, posing as the sender. The other way to launch a replay attack is, as shown in Figure 1.1(c). A replay attack has a two-fold consequence. First, the attacker is replaying the message of another node. Second, the attacker is transmitting stale information. In particular, the chances of the information being stale are higher in networks with higher node mobility. When replay attacks are launched on the localization process, a localizing node will receive an incorrect reference thereby localizing incorrectly. Unlike a wormhole attack, a single node can disrupt the network with a replay attack [1, 8].

- **Sybil Attack:** The Sybil attack requires a more sophisticated attacker compared to the replay attack. In a Sybil attack, a node claims multiple identities in the network. When launched on localization, localizing nodes can receive multiple location references from a single node leading to incorrect location estimation. Like the replay attack, the Sybil attack can also be launched by a single node since there is no need for collusion among nodes to launch this attack [1].

- **Wormhole Attack:** A wormhole attack is the most complicated of all the mentioned attacks. To launch a wormhole attack, the attacker has to compromise at least two nodes. In The colluding nodes in the network, tunnel messages are transmitted in one part of the network to their colluding partners in other parts of the network. The effect of a wormhole attack on localization is depicted in Figure 1.1(d). Here, node *A* is sending its reference to nodes *B* and *C*. However, since there is a wormhole link between *C* and *G*, *G* can locally replay the location reference of *A* in its neighborhood, misleading node *F*. Consequently, *F* will compute its location incorrectly. Intuitively, wormhole attacks pose more serious problems in range-free localization compared to range-based localization [1].

Chapter 2

EXISTING SECURE LOCATION DISCOVERY SCHEMES

Several techniques have been developed to deal with the security problems of location discovery in wireless sensor networks. The location verification technique proposed in Secure Verification of Location Claims [14] can be used to verify the relative distance between a verifying node and a sensor node. However, it does not provide a solution to conduct secure location estimation at non-beacon nodes. A robust location detection is developed in Robust Location Detection [13] using the idea of majority voting, but it cannot be directly applied in resource constrained sensor networks due to its high computation and storage overheads. A robust statistical method is independently discovered in Robust Statistical Method [7] to achieve robustness through Least Median of Squares.

SeRLoc [5] protects location discovery with the help of sectored antennae at beacon nodes. Similar to the voting-based scheme, SeRLoc can tolerate malicious attacks by adopting the idea of majority voting. SPINE [2] was developed to protect location discovery by using verifiable multi-lateration. However, the distance bounding techniques required for verifiable multi-lateration may not be available in sensor networks due to the difficulties in dealing with the external attacks in ultrasound-based distance bounding and achieving nanosecond processing and time measurements in radio-based distance bounding. ROPE [6] is developed by integrating SeRLoc and SPINE. However, it still requires nanosecond processing and time measurements that are not desirable for the current generation of sensor networks.

MMSE [8, 9] is a recently developed scheme that deals with malicious attacks against location discovery using statistical approach, where minimum mean square is used as an indicator to identify and remove the inconsistent malicious location references. In [8, 9], they have also developed a secure discovery scheme that adopts an iteratively refined voting scheme to tolerate malicious location references introduced by attackers. Both statistical and voting based approaches are purely based on a range based location discovery scheme, where the location references may come from beacon nodes that are either single hop or multiple hops away, or from those non-beacon nodes that already estimated their location [9]. In this thesis, the work will be focused on voting based location discovery scheme.

2.1 Statistical Based Location Discovery Scheme:

Intuitively, a location reference introduced by a malicious attack is aimed at misleading a sensor node about its location. Thus, it is usually different from benign location references. When there are redundant location references, there must be some inconsistency between the malicious location references and the benign ones (an attacker may still have a location reference consistent with the benign ones after changing both the location and the distance values. However, such a location reference will not generate significantly negative impact on location determination). To take advantage of this observation, the inconsistency among the location references are used to identify the malicious ones, and discard them before finally estimating the locations at sensor nodes [9].

In a statistical based scheme, the sensor node uses a Minimum Mean Square Error (MMSE) based method to estimate its own location. Thus, most current range-based localization methods can be used with this technique. To harness this observation, the sensor's location is first estimated with the MMSE-based method and then assessed if the estimated location could be derived from a set of consistent location references. If so, the estimation result is accepted; otherwise, the most inconsistent location references are identified and removed, and the same process is repeated. This process may continue until a set of consistent location references is found or it is not possible to find such a set [9].

2.1.1. Checking the consistency of location references

The mean square error ς^2 of the distance measurements based on the estimated location is used as an indicator of the degree of inconsistency, since all the MMSE-based methods estimate a sensor node's location by (approximately) minimizing this mean square error [9].

Definition 1: Given a set of location references $L = \{\langle x_1, y_1, \delta_1\rangle, \langle x_2, y_2, \delta_2\rangle, \cdots, \langle x_m, y_m, \delta_m\rangle\}$ and a location $(\hat{x}, \hat{y})$ estimated based on L, the mean square error of this location estimation is:

$$\varsigma^2 = \sum_{i=1}^{m} \frac{\left(\delta_i - \sqrt[2]{(\hat{x} - x_i)^2 + (\hat{y} - y_i)^2}\right)^2}{m} \qquad (2.1)$$

Intuitively, the more inconsistent a set of location references is, the greater the corresponding mean square error should be. A simple, threshold-based method is used to determine if a set of location references obtained at a sensor node is τ-consistent with

respect to a MMSE-based method if the method gives an estimated location $(\hat{x}, \hat{y})$ such that the mean square error of this location estimation is [9]:

$$\varsigma^2 = \sum_{i=1}^{m} \frac{\left(\delta_i - \sqrt[2]{(\hat{x} - x_i)^2 + (\hat{y} - y_i)^2}\right)^2}{m} \leq \tau^2 \qquad (2.2)$$

2.1.2. Identifying the largest consistent set

Since the MMSE-based methods can deal with measurement errors better if there are more benign location references, as many benign location references should be kept as possible while the malicious ones are removed. This implies the largest set of consistent location references should be achieved.

Brute-force Algorithm (BARMMSE): Given a set *L* of *n* location references and a threshold *τ*, a simple approach to computing the largest set of *τ*-consistent location references is to check all subsets of *L* with *i* location references about *τ*-consistency, where *i* starts from *n* and drops until a subset of *L* is found to be *τ*-consistent or it is not possible to find such a set. Thus, if the largest set of consistent location references consists of *m* elements, a sensor node has to use the MMSE method at least $1 + \binom{n}{m+1} + \cdots + \binom{n}{n}$ times to find the right one. If $n = 10$ and $m = 5$, a node needs to perform the MMSE method for at least 387 times. It is certainly not desirable to do such expensive operations on resource constrained sensor nodes [9].

Greedy Algorithm (GARMMSE): To reduce the computation on sensor nodes, a greedy algorithm can be used, which is simple but suboptimal. This greedy algorithm works in rounds. It starts with the set of all location references in the first round. In each round, it first verifies if the current set of location references is *τ*-consistent. If so, the algorithm outputs the estimated location and stops. Optionally, it may also output the set of location references. Otherwise, it considers all subsets of location references with one fewer location reference, and chooses the subset with the least mean square error as the input to the next round. This algorithm continues until it finds a set of *τ*-consistent location references or when it is not possible to find such a set (i.e., there are only three remaining location references) [9].

The greedy algorithm significantly reduces the computational overhead in sensor nodes. To continue the earlier example, a sensor node only needs to perform MMSE operations about 50 times (instead of 387 times) using this algorithm. In general, a sensor node needs to use a MMSE-based method for at most $1 + n + (n-1) + \cdots + 4 = 1 +$

$\frac{(n-3)(n+4)}{2}$ times. However, the greedy algorithm cannot guarantee that it can always identify the largest consistent set. It is possible that benign location references are removed, which generates a big impact on the accuracy of location estimation, especially when there are multiple malicious location references. To deal with this problem, an enhanced greedy algorithm was developed based on an efficient approach to identify the most suspicious location reference from a set of location references [9].

Enhanced Greedy Algorithm (EARMMSE): In the previous discussion, only the consistency of 3 or more location references was considered. A further investigation also reveals that two benign location references are usually consistent with each other such that there exists at least one location in the deployment field on which both location references agree. Hence, when the majority of location references are benign, many location references can usually be found so that they are consistent with a benign location reference. In addition, when a malicious location reference tries to create a larger location error, the number of location references that are consistent with the malicious one will decrease quickly [9].

According to the above discussion, for each location reference the number of location references that are consistent with this location reference can simply be counted. This number is called the degree of consistency and can be used to rank the suspiciousness of the location references received at a particular non-beacon node. The smaller the degree is, the more likely that the corresponding location reference is malicious [9].

The consistency between two location references can be verified as follows. For any location reference $\langle x, y, \delta \rangle$, the non-beacon node derives the area that it may reside based on this location reference. This area can be represented by a ring centered at (x, y), with the inner radius max {δ- ϵ, 0} and the outer radius (δ+ ϵ), where ϵ is the maximum distance error. For the sake of presentation, such a ring is referred to as the candidate ring (centered) at location (x, y). The non-beacon node then checks whether the candidate rings of two location references overlap each other. If the candidate rings overlap, they are consistent; otherwise, they are not consistent [9].

The algorithm to check whether the candidate rings of two location references, $a = \langle x_a, y_a, \delta_a \rangle$ and $b = \langle x_b, y_b, \delta_b \rangle$, overlap can be done efficiently in the following way: Let d_{ab} denote the distance between (x_a, y_a) and (x_b, y_b) and let $r_{max}(x)$ and $r_{min}(x)$ denote the outer radius and the inner radius of the candidate ring of location reference x, respectively. We can easily figure out that the candidate rings of location references a and b will not overlap when any of the following three conditions are true:

(1) $d_{ab} > r_{max}(a) + r_{max}(b)$,

(2) $d_{ab} + r_{max}(a) < r_{min}(b)$ and

(3) $d_{ab} + r_{max}(b) < r_{min}(a)$.

Similar to the greedy algorithm, the enhanced algorithm has to identify the largest consistent set starting with the set of all location references in the first round. In each round, it verifies whether the current set of location references is τ-consistent. If the current set is τ-consistent, the algorithm outputs the estimated location and stops. Optionally, it may also output the set of location references. Otherwise, the algorithm removes the location reference corresponding to the smallest degree and uses the remaining location references as the input to the next round. This algorithm continues until it finds a set of τ-consistent location references or when it is not possible to find such a set (i.e., there are only three remaining location references) [9].

The enhanced algorithm not only improves the accuracy of location estimation in the presence of malicious attacks, but also reduces the computation overhead significantly since it can identify the most suspicious location reference efficiently and effectively. To continue the earlier example, a non-beacon node only needs to perform MMSE operations five times. In general, a non-beacon node needs to use a MMSE-based method, at most, n-3 times [9].

2.2 Voting Based Scheme

In this approach, each location reference votes on the locations at which the node of concern may reside. To facilitate the voting process, the target field is quantized into a grid of cells, and each sensor node determines how likely it is in each cell based on each location reference. The cell(s) with the highest vote is selected and the center of the cell(s) used as the estimated location. To deal with the resource constraints on sensor nodes, an iterative refinement scheme is developed to reduce the storage overhead, improve the accuracy of estimation, and make the voting scheme efficient on resource constrained sensor nodes [9].

2.2.1. The basic scheme

After collecting a set of location references, a sensor node determines the target field. The node does so by first identifying the minimum rectangle that covers all the locations declared in the location references, and then extending this rectangle by R_b, where R_b is the maximum transmission range of a beacon signal. This extended rectangle forms the target field, which contains all possible locations for the sensor node. The sensor node then divides this rectangle into M small squares (cells) with the same side length L, as

illustrated in Figure 2.1. (The node may further extend the target field to have square cells.) The node then keeps a voting state variable for each cell, initially set to 0.

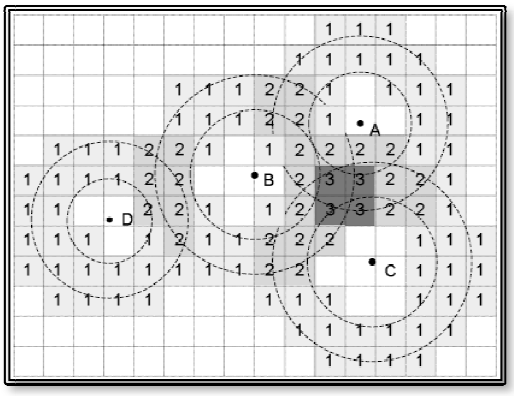

Figure 2.1: The voting based location estimation [9].

At the beginning of this algorithm, the non-beacon node needs to identify the candidate ring of each location reference. For example, in Figure 2.1, the ring centered at point *A* is a candidate ring at *A*, which is derived from the location reference with the declared location at *A*. For each location reference , the sensor node identifies the cells that overlap with the corresponding candidate ring and increments the voting variables for these cells by 1. After the node processes all the location references, it chooses the cell(s) with the highest vote, and uses its' (their) geometric centroid as the estimated location of the sensor node [9].

2.2.2. Iterative refinement

The number of cells *M* (or equivalently, the quantization step *L*) is a critical parameter for the voting-based algorithm. It has several implications to the performance of this approach. First, the larger *M* is, the more state variables a sensor node has to keep, and thus the more storage required. Second, the value of *M* (or *L*) determines the precision of location estimation. The larger *M* is, the smaller each cell will be. As a result, a sensor node can determine its location more precisely based on the overlap of the cells and the candidate rings [9].

However, due to the resource constraints on sensor nodes, the granularity of the partition is usually limited by the memory available for the voting state variables on the nodes. This puts a hard limit on the accuracy of location estimation. To address this problem, an iterative refinement is proposed to the above basic algorithm to achieve fine accuracy with reduced storage overhead. In this version, the number of cells *M* is chosen according to the memory constraint in a sensor node. After the first round of the

algorithm, the node may find one or more cells having the largest vote. To improve the accuracy of location estimation, the sensor node then identifies the smallest rectangle that contains all the cells having the largest vote, and performs the voting process again. For example, in Figure 2.1, the same algorithm will be performed in a rectangle which exactly includes the four cells having three votes [9].

By having a smaller rectangle to quantize in a later iteration, the size of cells can be reduced, resulting in a higher precision. Moreover, a malicious location reference will most likely be discarded, since its candidate ring usually does not overlap with those derived from benign location references. For example, in Figure 2.1, the candidate ring centered at point *D* will not be used in the second iteration [9].

The iterative refinement process should terminate when a desired precision is reached or the estimation cannot be refined. The former condition can be tested by checking if the side length L of each cell is less than a predefined threshold S, while the latter condition can be determined by checking whether L remains the same in two consecutive iterations. The algorithm then stops and outputs the estimated location obtained in the last iteration. It is easy to see that the algorithm will fall into either of these two cases, and thus will always terminate [9].

Chapter 3

PROPOSED SECURE LOCATION DISCOVERY SCHEMES

The main focus of this thesis work was focused on the voting-based scheme developed in [8, 9], because it is more adaptive to be implemented on range-free location discovery schemes, whereas the statistical MMSE scheme developed in [8, 9] can only work on range-based location discovery schemes. The work took two phases, first phase was to enhance voting-based location discovery scheme to be more applicable on wireless sensors without the usage of distance measurements, such as ToA or RSSI. The second phase was to develop a secure range-free location discovery scheme inspired by the enhanced voting-based technique.

3.1 Enhanced Voting Based Scheme

In sensor networks and other distributed systems, errors can often be masked through fault tolerance, redundancy, aggregation, or by other means. Depending on the behavior and requirements of protocols using location information, varying granularities of error maybe appropriate from system to system. Acknowledging that the cost of the hardware required by range-based solutions maybe inappropriate in relation to the required location precision, researchers have sought alternative range-free solutions to the location discovery problem in sensor networks. These range-free solutions use only regular radio modules as basics for location discovery; hence, they do not incur any additional hardware cost [18].

This argument gives inspiration to use other means to measure the distance between the sensor node and its location references without the need for the hardware associated with distance measurement, such as RSSI or ToA. In homogeneous wireless sensor networks, where all sensor nodes have common radio coverage range, a good distance measure would be the combination of radio range and the number of hop counts between the sensor and the location references sources. The distance can be estimated as the product of the radio coverage range of the sensor node by the number of hop counts between the sensor node and the location reference source, as shown in Figure 3.1.

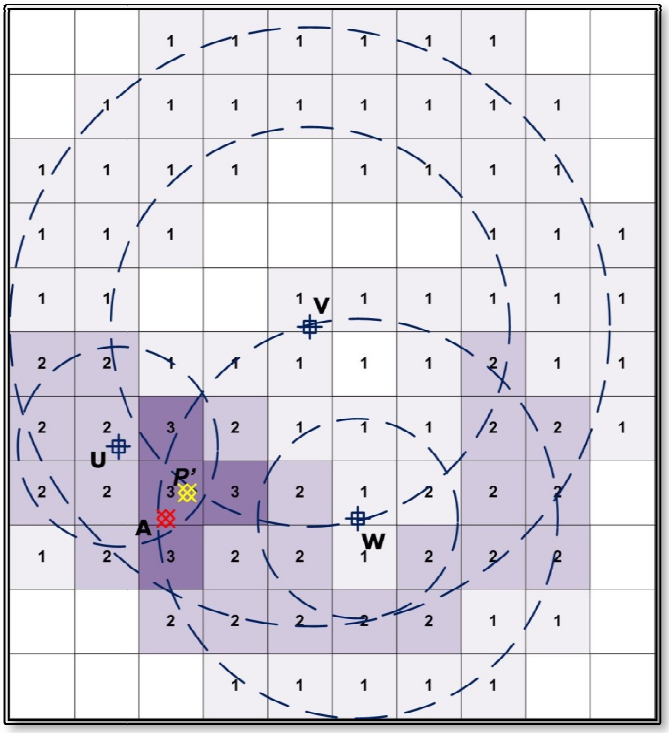

Figure 3.1: *Enhanced Voting-Based Scheme.*

The sensor A receives three location references from sensors: U, V and W which are approximately one, three and two hops away from sensor A, respectively. Sensor A would perform the enhanced voting based location discovery to estimate its location at point p'.

The enhanced voting-based location discovery scheme can be initiated at the sensor node once it receives enough location references to estimate its location. In practice, receiving a minimum of three location references by a sensor node enables it to trigger a location discovery process. The location references would be used to get the extent of the deployment map for that sensor, so that it contains all the location references received by the sensor node. Then the deployment map is quantized into small square cells, where the sensor node would initialize the voting state of those cells to zero.

The voting process starts by identifying the candidate rings for each location reference. The candidate rings for a location reference consist of two rings, an inner ring with radius of Max[radio range × (hop count − 1), 0], and an outer ring of radius (radio range × hop count). The second step in the voting process is to mark the cells that overlap with the corresponding candidate rings of each location reference and increment their vote by 1. Checking the overlap of candidate rings and cells for a given location reference is illustrated in Figure 3.2.

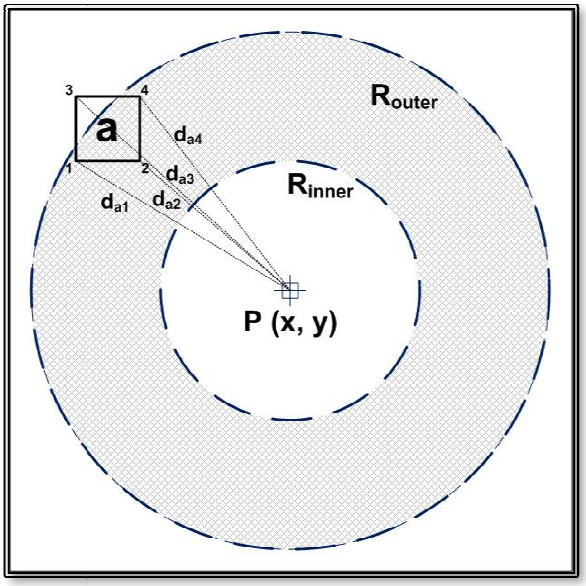

Figure 3.2: *Overlapping of candidate rings and a cell.*

With the intention of checking to see if a given cell, such as cell *a*, overlaps with the candidate rings of reference point P, let $d_{a1}(P)$, $d_{a2}(P)$, $d_{a3}(P)$ and $d_{a4}(P)$ denote the distances between point P and the vertices of cell *a*, respectively. The candidate rings overlap with cell *a* if and only if the distance between point P and any of the cell *a* vertices fall between the candidate rings of point P:

Where:

R_{inner} is the inner candidate ring for location reference point P.

R_{outer} is the outer candidate ring for location reference point P.

$d_{ai}(P)$ is the distance between point P and vertex *i* of cell *a*.

After the node processes all the location references, it selects the cells with highest vote and uses their geometrical centroid as the estimated location of the sensor node. The enhanced voting based scheme also supports iterative refinement as in the original scheme presented in [8, 9]. The cells with maximum vote will be quantized further into smaller cells and fed back into the voting process algorithm again to refine the estimated location results.

The enhanced voting based location discovery scheme can be summarized in the following pseudo code:

- *Receive location references {(X_1, Y_1, $Hops_1$), (X_2, Y_2, $Hops_2$), ...};*
- *Determine the extents of deployment field:*
- *Lower left corner = min. x-coordinate and y-coordinate values among the location reference;*
- *Upper right corner = max. x-coordinate and y-coordinate values among the location reference;*
- *Quantize the deployment field into small square cells;*
- *For each cell in quantized deployment field, initialize the voting value to 0;*
- *For each location reference (X_i, Y_i, $Hops_i$) {*
  - *set innerRing radius = min[0, radioRange×($hops_i$-1)]*
  - *set outerRing radius = radioRange × $hops_i$*
  - *for each cell in quantized deployment field {*
    - *if (distance (any cell vertex, (X_i, Y_i)) >= innerRing && <= outerRing)*
      increment cell vote by 1;

  }

}
- *highVoteCells = Get the set of cells with highest vote;*
- *estimatedLocation = Compute the geometrical centroid of highVoteCells;*
- *return estimatedLocation;*

3.2 Secure Range-Free Location Discovery Scheme

Most of the security schemes presented so far focused on range-based location discovery schemes. The main thesis goal is to develop a secure range-free location discovery scheme inspired by voting-based scheme. The enhanced voting-based scheme will be applied to APIT [18] which is a range-free location discovery scheme, but does not have consideration of resilience against location discovery attacks.

3.2.1 Area-based Point In Triangle location discovery scheme (APIT)

APIT employs a novel area-based approach to perform location estimation by isolating the environment into triangular regions between beacon nodes, as shown in Figure 3.3. Nodes inside or outside of these triangular regions allow a node to narrow down the area in which it can potentially reside. By utilizing combinations of beacon nodes positions, the diameter of the estimated area where a node resides can be reduced to provide good location estimation [18].

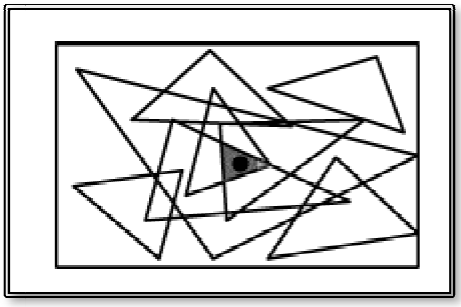

Figure 3.3: *APIT location discovery scheme [18].*

The theoretical method used to narrow down the possible area in which a target node resides is called the Point-In-Triangulation test (PIT). In this test, a node chooses three location references and tests whether it is inside the triangle formed by connecting these three references. APIT repeats this PIT test with different location references combinations until all combinations are exhausted or the required accuracy is achieved. At this point, APIT calculates the Center of Gravity (COG) of the intersection of all the triangles in which a node resides to determine its estimated position [18].

The APIT algorithm can be broken down into four steps:

1. Beacon exchange
2. PIT testing
3. APIT aggregation
4. COG calculation

These steps are performed at individual sensor nodes in a purely distributed fashion. The pseudo code for APIT algorithm is described below:

- *Receive location references (X_i, Y_i) from N beacon nodes.*
- *InsideSet = □ // the set of triangles in which the sensor resides*
- *For (each triangle Ti triangles) {*
  - *If (Point-In-Triangulation test (Ti) == true)*
    InsideSet = InsideSet {Ti};
  - *If (accuracy(InsideSet) > enough)*
    Break;

  },

  / Center of gravity (CoG) calculation */*
- *Estimated Position = CoG (Ti InsideSet);*

The size of InsideSet is noticeably growing cubically with the number of location references received. For example, with 30 location references in a sensor network of 1500 sensor nodes, the radio region will be divided by 4060 triangles into small pieces. If the PIT tests render correct inside/outside decisions, each decision will narrow down the area in which a target node can possibly reside, making the final error small [18].

3.2.2 Creating triangles from location references

The major issue in developing a secure range-free location discovery scheme, which combines the security features of voting-based scheme [8, 9] and range-free estimation of APIT [18], is how to generate the triangle areas that will divide the deployment field into small regions. The regions that represent the highest intersection of those triangular areas will contain the estimated location of the sensor node.

In APIT, a triangle is formed by any three location references chosen at random. The sensor node exhausts all the possible combinations of triangles formed by the location references that it received until it can find a combination of triangles that gives a maximum intersectional region [18]. This process is very slow and might not provide the best accuracy required in the estimated location. Also, it does not verify the credibility of the location references in regard to whether they are legitimate or compromised.

In range-free location discovery, the sensor node has no distance or direction measurements between itself and the location references that it receives; therefore, the sensor cannot predict inside which triangle it resides. Instead, the sensor node can have a rough estimation of its location with respect to the location references using the number of hops and radio transmission range. Figure 3.4, shows the method developed in this thesis to create triangular regions that contain the estimated location of the senor node.

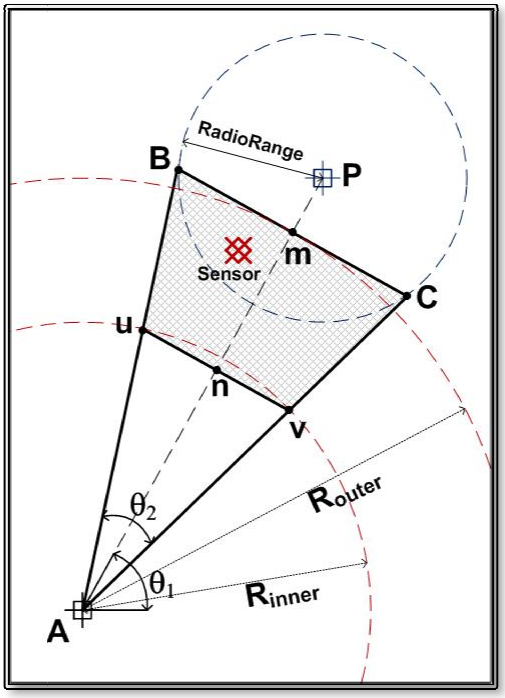

Figure 3.4: *Triangle creation method.*

The sensor node picks the location reference with the minimum number of hops (the closest reference to this sensor), point *P* in this example, and uses this location reference as a guide point to direct the triangles toward it. Each location reference represents a triangle vertex, point *A* in this example, and the base of the triangle will be directed towards the guide point, as shown in Figure 3.4. The triangle vertices can be defined using simple geometry and trigonometric functions as described in the equations below:

To increase the accuracy of estimated location of the sensor node, the triangle includes a smaller inner triangle that shares the same vertex defined by the location reference point. This provides the same effect as candidate rings in the enhanced voting-based scheme. As a result, the region at which the sensor node is most likely to reside is: Outer Triangle – Inner Triangle. The inner triangle vertices can be defined using the following equations:

3.2.3 Point-In-Triangulation test

There are several mathematical algorithms and equations that can check whether a point resides inside the triangle interior. In this thesis, the Barycentric Coordinates algorithm [19] was chosen. Barycentric coordinates are triples of numbers (t_1, t_2, t_3) corresponding to masses placed at the vertices of a reference triangle ΔABC. These masses then determine a point P, which is the geometric centroid of the three masses and is identified with coordinates (t_1, t_2, t_3), as shown in Figure 3.5.

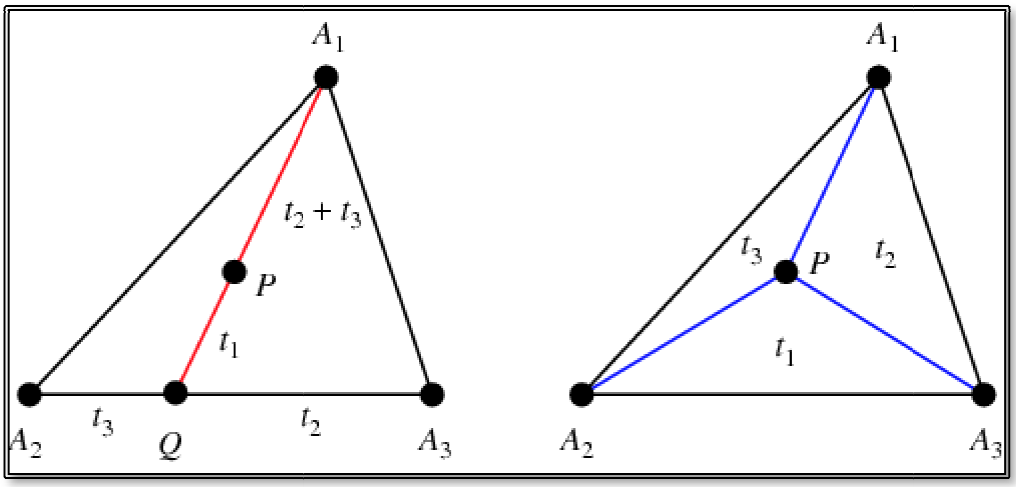

Figure 3.5: *Barycentric coordinates [19].*

Let the three points of the triangle define a plane in space, as shown in Figure 3.6. One of the points is chosen such that we all other locations on the plane can be considered as relative to that point. Next, basis vectors are needed to give coordinate values to all the locations on the plane. The two edges of the triangle that touch *A*: *(C−A)*

and *(B−A)* are selected. Any point on the plane can now be reached by starting at *A* walking some distance along *(C−A)* and from that point walking further in the direction *(B−A)* [15].

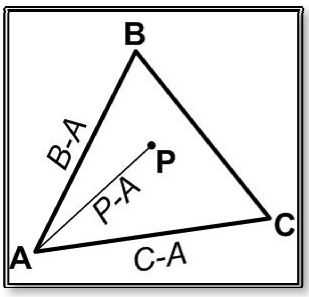

Figure 3.6: *Point in Triangle test [15]*.

With that in mind, we can describe any point on the plane as:

According to the above equation, point *P* can be inside the triangle *ABC* if and only if: $u, v > 0$, and $u + v$ 1.

Given the coordinates of point *P*, we can calculate *u* and *v* using the equation above to check whether the point *P* resides inside the triangle or not, as shown below [15]:

Having two unknown variables *u* and *v*, we need two equations to solve for them:

Solving for those two equations:

Where:

The point in triangle test can easily be programmed using the following pseudo code [15]:

```
- // Compute vectors
  w0 = (C−A)
  w1 = (B−A)
  w2 = (P−A)

- // Compute dot products
  dot00 = dot(w0, w0)
  dot01 = dot(w0, w1)
  dot02 = dot(w0, w2)
  dot11 = dot(w1, w1)
  dot12 = dot(w1, w2)

- // Compute barycentric coordinates
  invDenom = 1 / (dot00 × dot11 − dot01 × dot01)
  u = (dot11 × dot02 − dot01 × dot12) × invDenom
  v = (dot00 × dot12 − dot01 × dot02) × invDenom

- // Check if point is in triangle
  return (u > 0) && (v > 0) && (u + v < 1)
```

3.2.4 The proposed secure range-free location discovery algorithm

Putting the triangle creation method and the point in triangle test together, the algorithm of a secure range-free location discovery is now defined. Using the example shown in Figure 3.7, the details of how this proposed algorithm works can be explained.

The sensor A receives three location references from sensors: U, V and W which are approximately one, three and two hops away from sensor A, relatively. The location references would be used to get the extents of the deployment map for that sensor, so that it contains all the location references received by the sensor node. Then the deployment map is quantized into small square cells, where the sensor node initializes the voting state of those cells to zero.

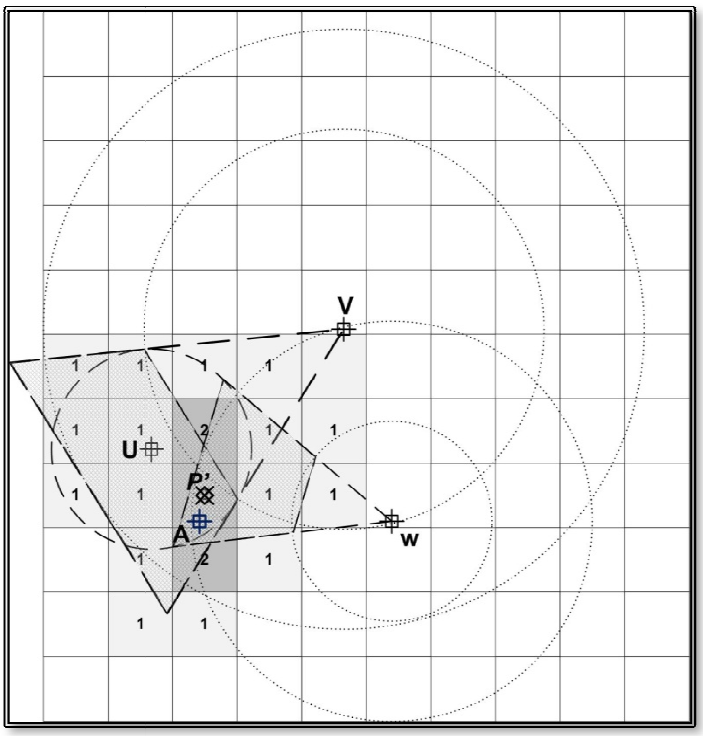

Figure 3.7: *Proposed secure range-free location discovery scheme.*

The voting process starts by identifying the guide point from the closest reference location received, in this example it would be point U. For each of the remaining location references, V and W, the inner and outer triangles would be created using the method described in section 3.2.b. The second step is the voting process of marking the cells that reside inside the region defined by subtraction of outer triangle from inner triangle (the shaded area shown in Figure 3.7) and incrementing their vote by 1. This is done by checking if any vertex of the cell is inside the shaded region using the Point-In-Triangle test described in section 3.2.c.

After the sensor node processes all the location references, it selects the cells with highest vote, and uses their geometrical centroid as the estimated location of the sensor node (point *P'* in our example). The secure range-free location discovery scheme can support iterative refinement as in the original scheme presented in [8, 9]. The cells with maximum vote will be quantized further into smaller cells and fed back into the voting process algorithm again to refine the estimated location results.

The proposed secure range-free location discovery scheme can be summarized in the following pseudo code:

- *Receive location references {(X_1, Y_1, $Hops_1$), (X_2, Y_2, $Hops_2$), ...};*
- *Determine the extents of deployment field:*
- *Lower left corner = min. x-coordinate and y-coordinate values among the location reference;*
- *Upper right corner = max. x-coordinate and y-coordinate values among the location reference;*
- *Quantize the deployment field into small square cells;*
- *Select the location reference with min. hops to be the guide point.*
- *For each cell in quantized deployment field, initialize the voting value to 0;*
- *For each location reference (X_i, Y_i, $Hops_i$) {*
  - *set outerTriangle (vertex=(X_i, Y_i);*
    Height=(radioRange × $Hops_i$);)
  - *set innerTriangle (vertex=(X_i, Y_i);*
    Side=(radioRange × ($Hops_i$-1));)
  - *for each cell in quantized deployment field {*
    - *perform PIT test on cell vertices;*
    - *if (PIT test returns TRUE for outerTriangle && FALSE for innerTriangle)*
      increment cell vote by 1;

  }

}
- *highVoteCells = Get the set of cells with highest vote;*
- *estimatedLocation = Compute the geometrical centroid of highVoteCells;*
- *return estimatedLocation;*

Chapter 4

IMPLEMENTATION AND TEST RESULTS

This chapter explores the implementation of both enhanced voting-based and secure range-free schemes on Sun SPOT wireless sensors. The schemes were tested on a combination of actual SPOT and virtual SPOT sensors using Solarium, the emulation environment provided by Sun SPOT. Another set of tests that focused on security resilience under various types of attacks was performed using a simulation program developed for this purpose.

4.1 Implementation on Sun SPOTs and System Architecture

4.1.1 Overview on Sun SPOT wireless sensors

The Sun SPOT sensor device is a small, wireless, battery powered experimental platform. It is programmed almost entirely in Java to allow programmers to create projects that require specialized embedded system development skills. The hardware platform includes a range of built-in sensors as well as the ability to easily interface to external devices [16].

A full, free-range Sun SPOT device is built by stacking a Sun SPOT processor board with a sensor board and battery, all packaged in a plastic housing, as shown in Figure 4.1. The smaller base-station Sun SPOT consists of only the processor board in a plastic housing. Each Sun SPOT has a 180MHz 32-bit ARM920T core processor with 512K RAM and 4M Flash. The Sun SPOT processor board has a 2.4GHz radio with an integrated on-board antenna. The radio is a TI CC2420 and is IEEE 802.15.4 compliant. There is no operating system used. The Sun SPOT runs a Java Virtual Machine directly on the hardware. The Sun SPOT uses a fully capable Java ME implementation, named Squawk which supports both CLDC 1.1 and MIDP 1.0, as well as providing basic OS functionality. The Java Virtual Machine executes directly out of flash memory. All of the device drivers are written in Java as well [16].

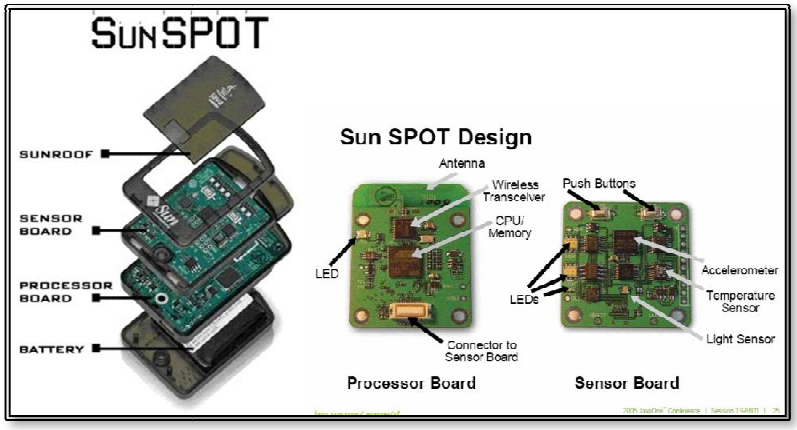

Figure 4.1: *Sun SPOT sensor [16].*

4.1.2 Implementation system architecture

Both the enhanced voting-based and secure range-free location discovery schemes are implemented in Java as library classes to be used in Sun SPOT wireless sensors. Each Sun SPOT runs an application that creates an instance of the secure location discovery library and sets the type of location discovery scheme to be used in the deployed sensor network. Figure 4.2 shows a block diagram of the system architecture at each Sun SPOT sensor.

The Sun SPOT sensor runs three main threads: receiving, transmitting and location discovery. These three threads run simultaneously as soon as the SPOT sensor is deployed. The receiving thread is responsible for keeping track of location reference messages from surrounding sensors and beacon nodes. Once the receiving thread gets a location reference message, it checks to see if this location reference is already included in the location reference list. If the location reference has already been received, then it is simply ignored, otherwise, the receiving thread retrieves the hops count and other routing information about the source of this location reference from the SPOT routing manager. The hops count is then attached with the location reference coordinates to be added to the location reference list.

The transmitting thread is responsible for broadcasting the SPOT's estimated location to its surrounding neighbors. The location discovery thread creates an instance of location estimator class, and initializes it with the basic information about the SPOT sensor (maximum radio range, location discovery scheme, initial quantization resolution). Once the SPOT sensor has received enough location references, the location discovery thread triggers the location discovery process and blocks both receiving and transmitting threads until the location is estimated.

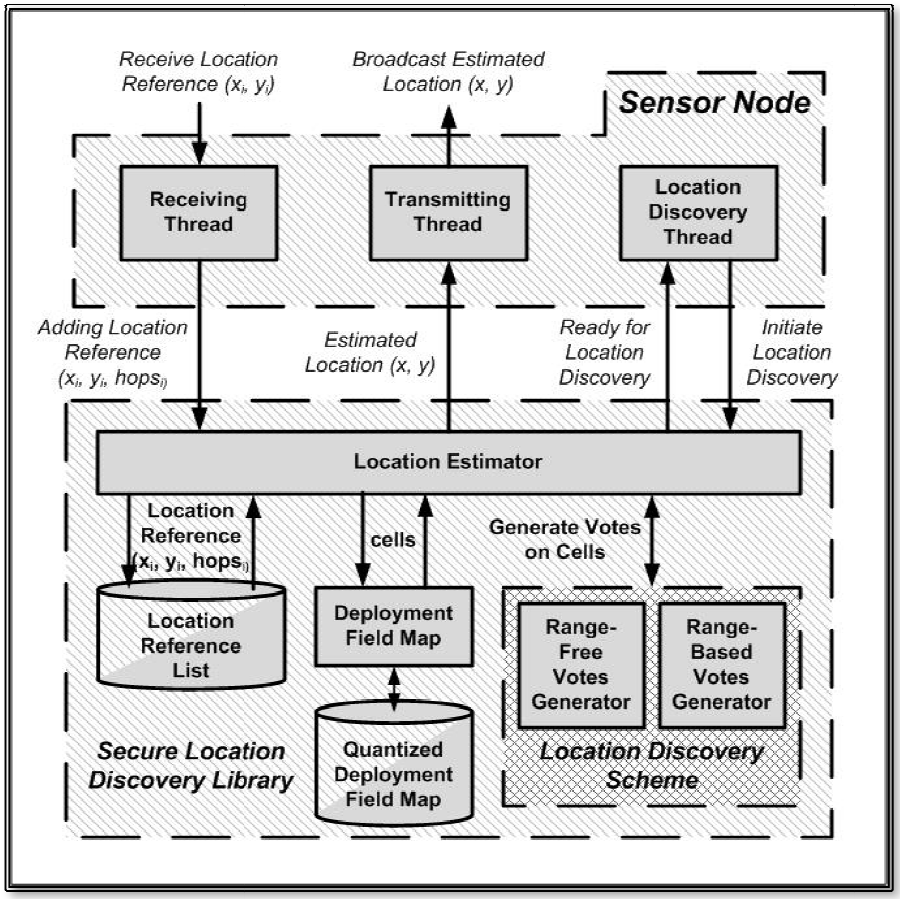

Figure 4.2: *System architecture block diagram.*

The secure location discovery library contains the necessary classes to perform either enhanced voting-based scheme, or range-free scheme. The main class in the library is the locationEstimator class which handles all the interfaces with the upper layer of threads.It also manages the location references, quantization of the deployment field map and cells vote generation. When a Sun SPOT creates an instance of the locationEstimator class it is initialized with the SPOT's radio coverage range, initial quantization resolution, and the type of location discovery scheme to be used to estimate the SPOT's location.

The locationEstimator class receives location reference from the receiving thread and stores those location references in the Location Reference List. When the locationEstimator receives enough location references (three or more reference since a minimum of three points are required to build a triangle), it sends a ready signal to the location discovery thread. The location discovery thread returns a request to the locationEstimator to initiate a location discovery process. The location discovery process starts by calculating the extent of the SPOT deployment field based on the received

location references. The deployment field map is then quantized into a set of cells, each with a zero vote value. The locationEstimator passes the list of location references and cells set to the location discovery scheme specified by the Sun SPOT. The location discovery scheme generates the votes on the cells set and returns the set to the locationEstimator, which in turn extracts the cells with maximum votes value and calculate the geometrical centroid (center of gravity for a mass) of those cells to be the estimated location.

Iterative refinement is performed by the locationEsitmator to increase the accuracy of the estimated location. This is done by quantizing the set of cells with maximum votes further into smaller sub-cells and feeding the new sub-cells back to the location discovery scheme for votes' generation. The process continues until the difference between two consecutive estimated locations are less than 5%. At this point, the locationEstimator returns the estimated location to the location discovery thread to be broadcasted by the transmitting thread.

4.2 Test Cases and Simulation Results

This thesis presents two sets of test cases. The first is implemented on Sun SPOTs wireless sensors to check the location discovery schemes accuracy and behavior. The second test case is implemented on a simulation program to investigate the security resilience of the location discovery schemes to various degrees of node compromise and location discovery threats.

4.2.1 Test on Sun SPOTs and Solarium

The first experiment is to test the accuracy of the estimated location of both the enhanced voting-based scheme and the secure range-free scheme by comparing the distance error between the sensors' actual location and their estimated location. The test was performed on Sun SPOT wireless sensors using the Solarium emulator. Solarium is a Java™ application that can be used to remotely manage a network of Sun SPOTs. With Solarium, SPOTs can be discovered and the life-cycle of the applications running on those devices can be managed [17].

Solarium includes an emulator capable of running a Sun SPOT application on a desktop computer. This allows for testing a program before deploying it to a real SPOT, or if a real SPOT is not available. Instead of a physical sensor-board, virtual SPOTs have a sensor panel that is used to set any of the potential sensor inputs (e.g. light level, temperature, digital pin inputs, analog input voltages, and accelerometer values). The application controls the resources available in the virtual SPOT, just as in a real SPOT.

Receiving and sending via the radio is also supported. Each virtual SPOT is assigned its own address and can broadcast or unicast to the other virtual SPOTs. If a shared basestation is available, a virtual SPOT can also interact over radio with real SPOTs [17].

When a virtual SPOT is created in Solarium, a new process is started to run the emulator code in a Squawk Virtual Machine (VM). The emulator code communicates over a socket connection with the virtual SPOT GUI code in Solarium. For example when the SPOT application changes the RGB value of an LED, that information is passed to the virtual SPOT GUI code which updates the display for that LED with the new RGB value. Likewise when the user clicks on one of the virtual SPOT's switches using the mouse, Solarium sends a message to the emulator code that the switch has been clicked, which can then be noticed by the SPOT application. Figure 4.3. shows a block diagram of the Emulator architecture [17].

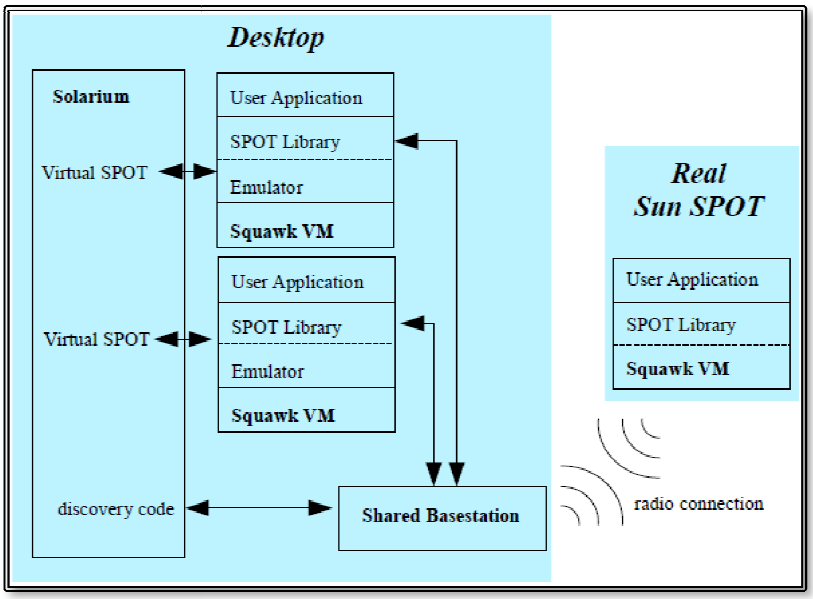

Figure 4.3: *Block diagram of emulator architecture in Solarium [17].*

Each virtual SPOT has its own Squawk VM running in a separate process on the host computer. Each Squawk VM also contains a complete host-side radio stack as part of the SPOT library. This allows the SPOT application to communicate with other SPOT applications running on the host computer, such as other virtual SPOTs, using sockets or real SPOTs via radio if a shared basestation is running [17].

The experiment was implemented twice, once for each location discovery scheme, on a network of 9 virtual SPOTs, 3 of those SPOTs were beacon nodes and the rest were sensor nodes, as shown in Figure 4.4.

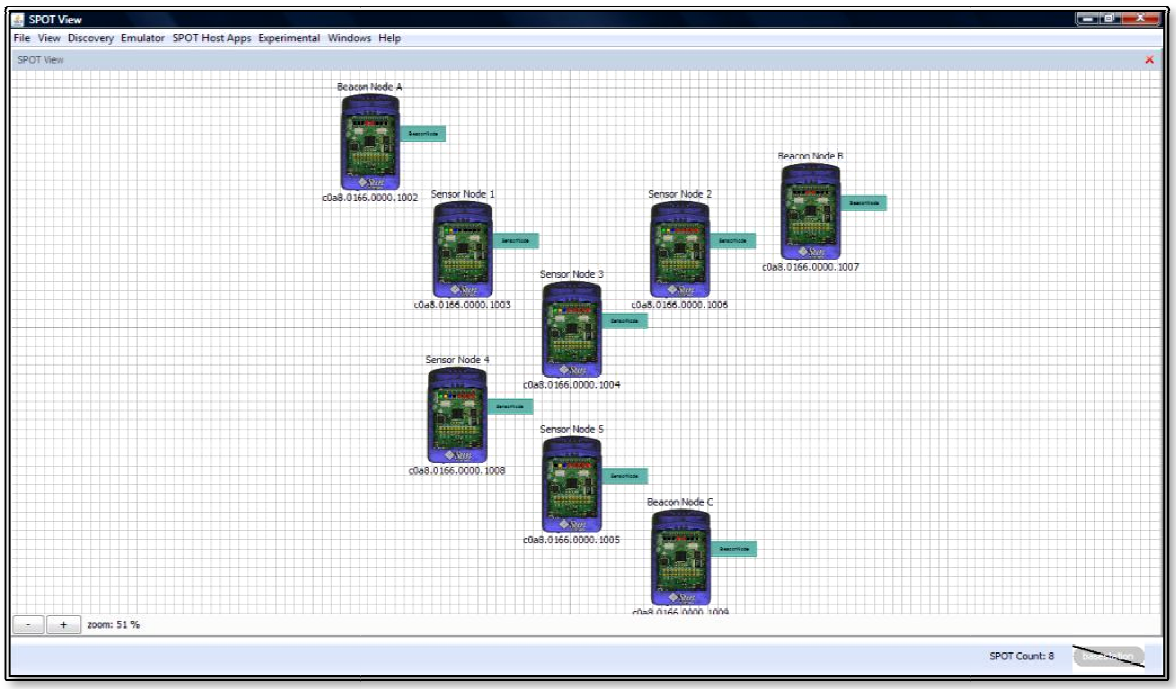

Figure 4.4: *Sun SPOTs experiment snapshot.*

The experiment starts with the beacon nodes broadcasting their locations to the other sensor nodes. Each sensor node receives location references from its neighbors and estimates its location accordingly. It then broadcasts the estimated location to the surrounding neighbors. After several rounds of exchanging location references and modifying the estimated location, each sensor node reaches a stabilized state where the changes in estimated locations are very small.

The results recorded for the Sun SPOT sensors, as shown in table 4.1, show that the estimated locations for both enhanced voting-based and secure range-free schemes were very close to the actual locations of their perspective sensor nodes. Figure 4.5 shows the Sun SPOTs layout map along with estimated location for each SPOT sensor.

Beacon Node		A		B		C	
Location		16.5, 42		49.5, 36		40.5, 9	
Sensor Node		1	2	3	4	5	
Actual Location		25.5, 36	40.5, 33	31.5, 27	20.5, 23	31.5, 15	
Estimated Location	Enhanced Voting-Based	26.14, 36.35	40.66, 29.91	30.42, 24.75	20.19, 21.62	32.86, 13.31	
	Secure Range-Free	26.14, 36.04	40.7, 30.55	33.6, 29.88	24.4, 22.8	32.58, 15.81	
Error Distance	Enhanced Voting-Based	0.73	3.09	2.5	1.41	2.17	
	Secure Range-Free	0.64	2.46	3.57	3.91	1.35	

Table 4.1: *Sun SPOTs experiment results.*

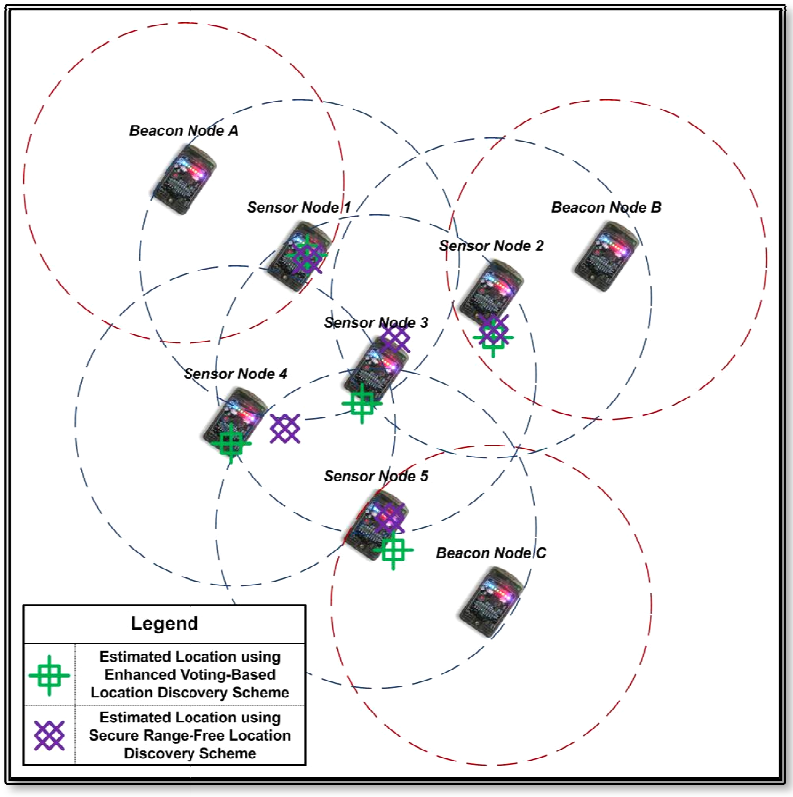

Figure 4.5: *Sun SPOTs experiment layout map and results.*

4.2.2 Simulation test and results

In order to test the security resilience of the proposed secure location discovery schemes, a test program was developed in Java to simulate a network of wireless sensor nodes subjected to various levels of security attacks. The test program contains three classes: Simulator, SensorNode, and Media, as shown in Figure 4.6. The Simulator class is the main class which is responsible for initializing the simulated sensor network and applying various security threats to that network while running the simulation. The SensorNode class represents a simulated sensor node which has an instance of the secure location discovery library (similar to a Sun SPOT sensor as described in section 4.1.b) and uses that library to initiate a location discovery process to estimate its location. The Media class represents the deployment field at which the simulated sensor network is operating. It contains three lists: beaconNodes, sensorNodes and compromisedNodes. Each list contains the sensor node actual location (as generated by the Simulation class) and its estimated location (as generated by the SensorNode class).

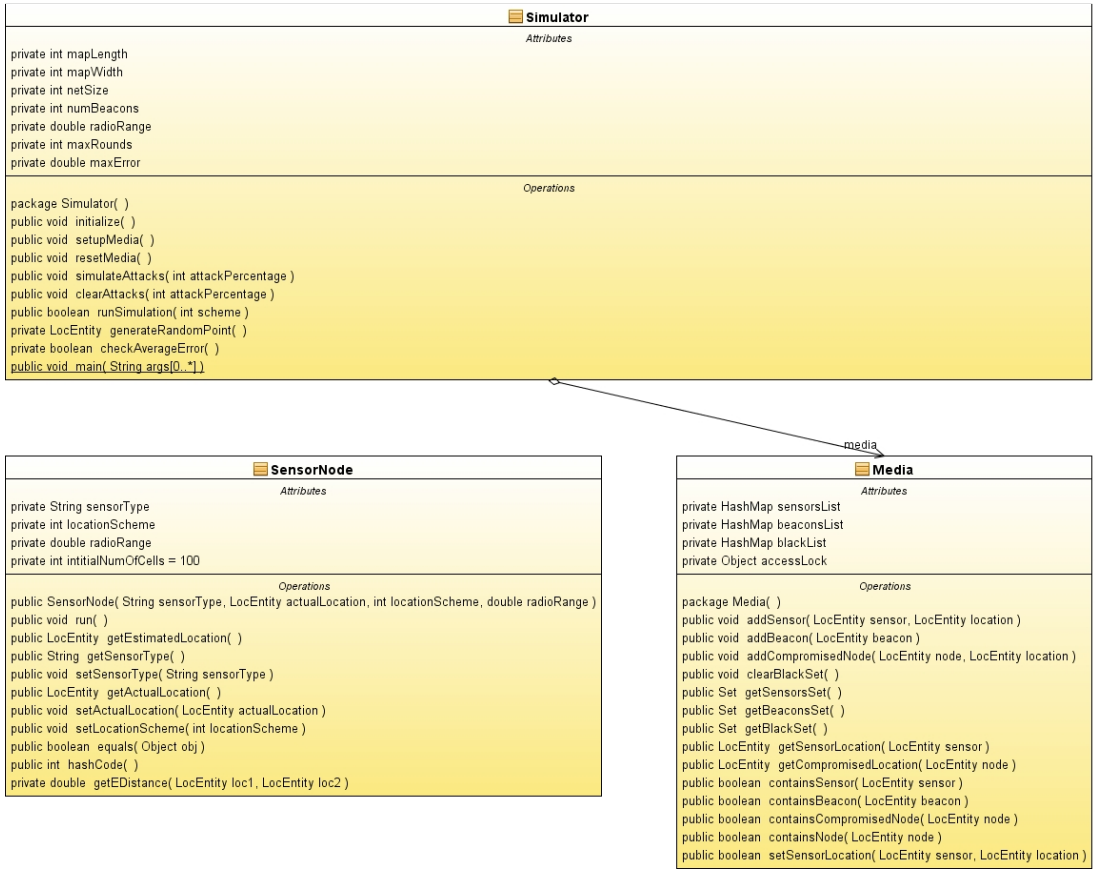

Figure 4.6: *Simulator class diagram.*

A test case starts by entering the parameters required by the Simulator class to simulate the sensor network. These parameters include the deployment field area, size of the simulated sensor network, number of beacon nodes in the simulated network, the radio coverage range for each sensor, maximum number of simulation rounds (so as not to run into infinite loops) and maximum acceptable error distance between a sensor node actual location and its estimated location. The Simulator class uses these parameters to generate and initialize the simulated sensor network, and then generates the location of each sensor randomly to fit within the simulation deployment field. Each simulated node is added to the Media class: beacon nodes are added to the beaconNodes list with their actual locations (as generated by the Simulator class) and sensor nodes are added to the sensorNodes list with their actual locations and setting the corresponding estimated locations to null value.

The simulation starts with no security threats (0% node compromise) and each simulated sensor initiates a first round of location discovery using enhanced voting-based scheme. The simulated sensors check their surrounding neighbors for location references and estimate their locations accordingly and store the results in the corresponding position in the sensorNodes list. After all the sensors finish estimating their locations, the average error distance and mean execution time are calculated for this round. The sensor nodes continue more rounds of location discovery to further increase the accuracy of their estimated locations until the average error distance becomes less than or equal to the maximum acceptable error distance for all the sensors in the network, or the number of simulation rounds reaches the maximum number of rounds.

At this point, the Simulator class resets the simulation network back to its initial state and begins the simulation process again for 0% node compromise, but this time using secure range-free scheme for location discovery. This process is repeated for 10%, 20%, 30%, 40% and 50% nodes compromise, and all the results of average error distance and average execution time are being recorded for each step.

The security threats are simulated through the Simulator class by choosing a number of sensors at random to act as malicious nodes. The number of those malicious nodes is specified by the attack percentage of node compromise. The attack pattern is a combination of compromised beacon nodes attack, reply attack and masquerade attack. The compromised beacon nodes attack is achieved by randomly removing some beacon nodes from the beaconNodes list and putting them into the compromisedNodes list where each compromised beacon node randomly generates a malicious location other than its actual location. The reply and masquerade attacks are achieved by randomly removing some sensor nodes from the sensorNodes list and putting them into the compromisedNodes list where each malicious sensor node randomly chooses a beacon node location to report as its own estimated location.

Five test cases were simulated using the test program for various network scales (25, 250, 500, 1000 and 1500 sensor nodes) with a maximum acceptable error distance of 1 unit distance, as shown in table 4.2. Each network size implements both the enhanced voting-based and secure range-free location discovery schemes. Both schemes were subjected to different levels of security threats and attacks, starting from 0% node compromise, up to 50% node compromise in intervals of 10%. The results of those five test cases were recorded as shown in Figure 4.7 and Figure 4.8.

Network Size	Deployment Field Area	Number of Beacon Nodes	Radio Coverage Range for each Sensor Node	Max. Number of Simulation Rounds	Max. Acceptable Error Distance
25	5 × 5	5	3	10	1
250	25 × 10	50	3	10	1
500	25 × 20	100	3	10	1
1000	50 × 20	200	3	10	1
1500	50 × 30	300	3	10	1

Table 4.2: *Simulation tests parameters.*

The simulation results, as shown in Figure 4.7, show that as the network size increases, the security resilience improves. This is clearly shown by the percentage of sensors that failed to estimate their location within the acceptable error distance tolerance. This percentage drops as the network size increases. This is due to the location references for each sensor node in the network increasing as the network size grows; this increases the accuracy of the estimated location for each sensor and provides more benign location references to withstand malicious references at various levels of attacks.

Comparing the results of enhanced voting-based scheme with secure range-free scheme, as shown in Figure 4.7, the enhanced voting-based scheme is more resilient than the secure range-free scheme. The percentage of sensor nodes that were unable to estimate their locations within the acceptable error distance tolerance using enhanced voting-based scheme were less than the percentage of sensor nodes unable to estimate their locations using secure range-free scheme. This is because in the secure range-free scheme each sensor node relies on its direct neighbors to be the guide points to estimate its location accurately. When some of those guide points are compromised, the accuracy decreases and the sensor node becomes more vulnerable to security attacks. Never the less, the resilience of secure range-free scheme was still good enough to withhold a security threat of 50% node compromise, as shown in Figure 4.7(b).

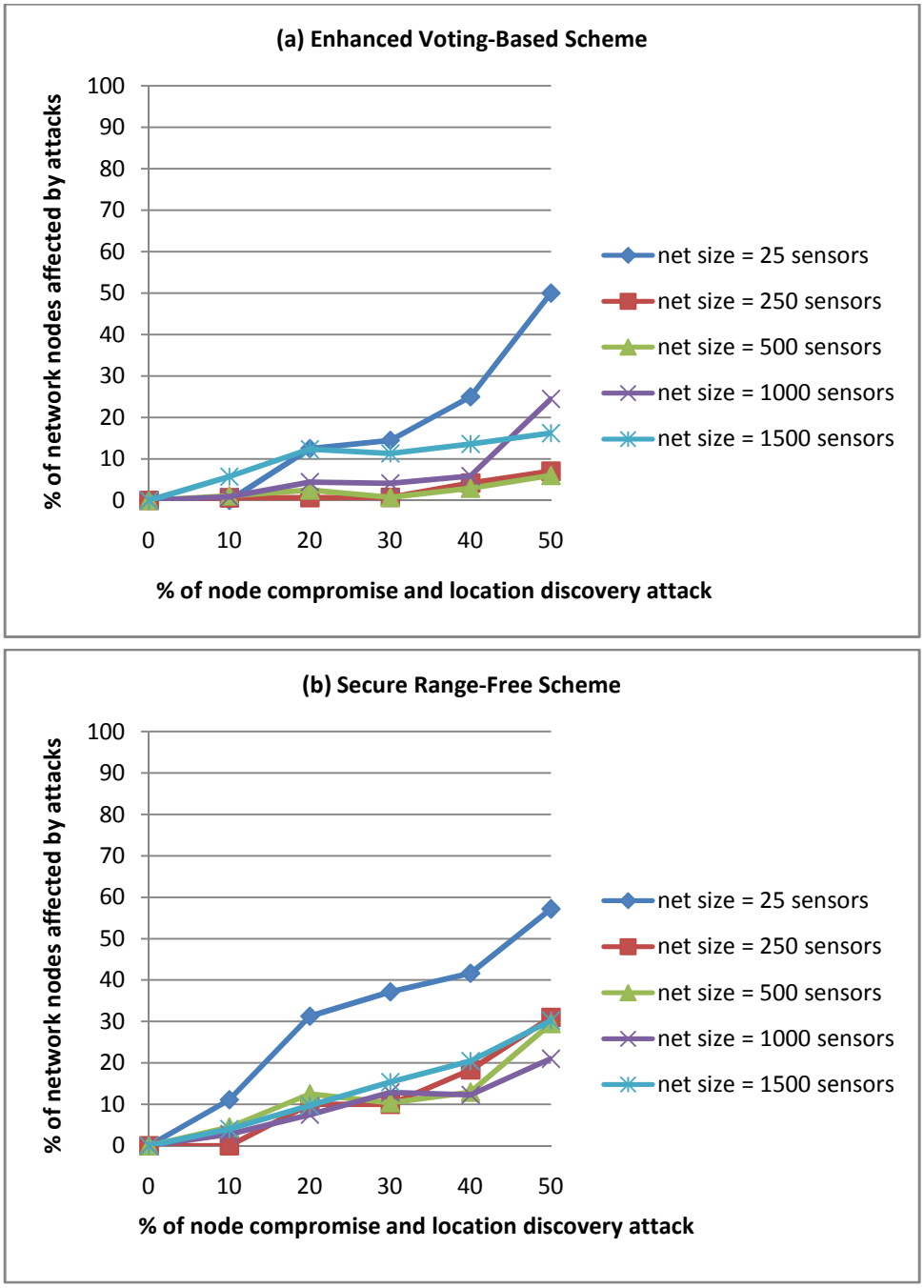

Figure 4.7: *Security resilience simulation test results.*

Figure 4.8 shows the average location discovery execution time. The enhanced voting-based scheme runs faster than the secure range-free scheme. This is because in secure range-free scheme, each sensor constructs a triangle from a received location reference towards every guide point, thus making the execution time a $O(n \times m)$, where n

is the number of location references and *m* is the number of guide points around the sensor node. Therefore, as the number of guide points increase the execution time increases as well. This is an important factor in increasing the accuracy of location discovery in the secure range-free scheme.

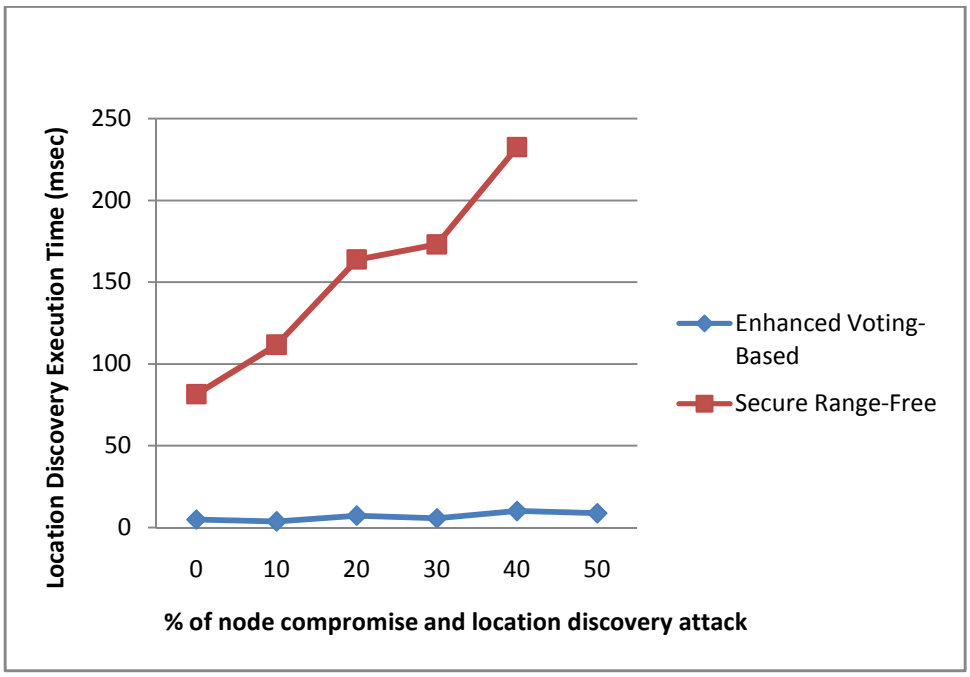

Figure 4.8: *Location discovery execution time.*

In order to optimize the execution time for secure range-free scheme, a comparison experiment was made on two networks of size 250 sensors. One network with the number of guide points for each sensor reduced to one guide point (picked randomly), and the other network has the number of guide points for each sensor set to maximum. The results of this experiment show improvement over execution time for the first network, where the time becomes an *O(m)*, as shown in Figure 4.9(b). Despite the improvement in execution time, a drawback in the security resilience for the first network occurs as shown in Figure 4.9(a). As the number of guide points decreases for a sensor node, in the case of an attack scenario, the probability of having malicious guide points increases. This makes the location discovery process more vulnerable.

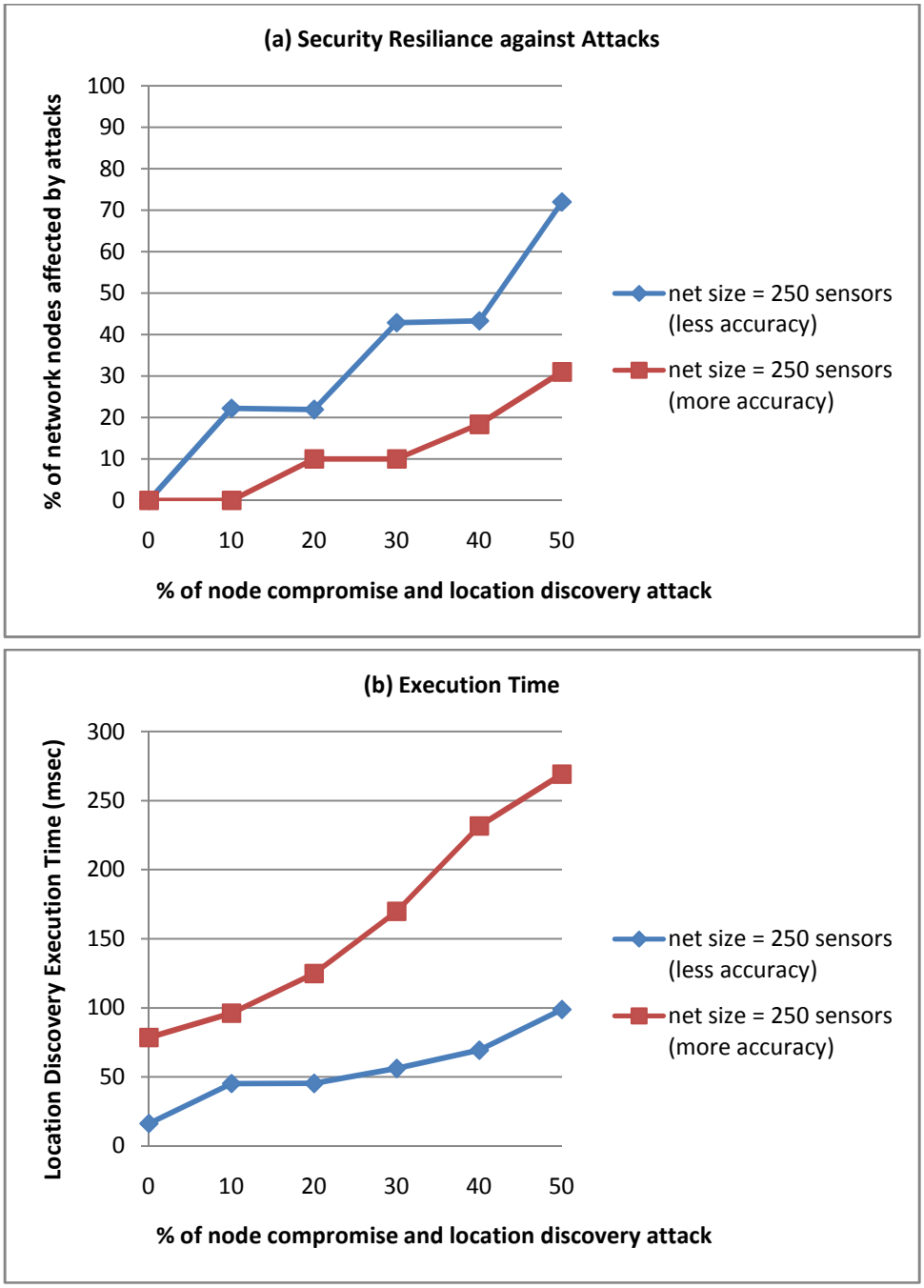

Figure 4.9: *Comparison test results.*

CONCLUSION

In this thesis, two location discovery schemes for wireless sensor networks are developed. The first scheme is an enhanced version of the voting based scheme developed in [8, 9] to be used for range-based location discovery. This enhanced scheme is used to inspire the development of a secure range-free location discovery scheme. The secure range-free scheme is tested and compared to the enhanced voting-based scheme in both field and simulation tests. Both schemes are implemented on Sun SPOT wireless sensors and tested for location discovery accuracy, giving good results in terms of estimated locations for each deployed SPOT sensor.

Simulation results show that both enhanced voting-based and secure range-free schemes are able to withstand security threats and location discovery attacks up to 50% of node compromise. Enhanced voting-based scheme showed resilience and accuracy under attack patterns more frequently than the secure range-free scheme. This is an inherited feature from range-based location discovery schemes. Range-free schemes tend to be less accurate than range-based schemes, yet more cost-effective [18]. The simulation results also show that optimizing the execution time for secure range-free scheme will affect its security resilience as well as location estimation accuracy.

REFERENCES

[1] A. Srinivasan and J. Wu, "A Survey on Secure Localization in Wireless Sensor Networks". Encyclopedia of Wireless and Mobile Communications, B. Furht (ed.), CRC Press, Taylor and Francis Group, 2008.

[2] Capkun, S. and Hubaux, J. "Secure positioning of wireless devices with application to sensor networks". In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05), 2005.

[3] D. Niculescu and B. Nath, "DV Based Positioning in Ad Hoc Networks". In Journal of Telecommunication Systems, 2003.

[4] He T., Huang C., Blum B., Stankovic J., and Abdelzaher T., "Range-Free Localization and Its Impact on Large Scale Sensor Networks". ACM Transactions on Embedded Computer Systems, Vol. 4, No. 4, November 2005.

[5] Lazos, L. and Poovendran, R. "Serloc: Secure range-independent localization for wireless sensor networks". In ACM Workshop on Wireless Security (WiSe'04). Philadelphia, PA, 2004.

[6] Lazos, L., Capkun, S., and Poovendran, R. "Rope: Robust position estimation in wireless sensor networks". In Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05), 2005.

[7] Li, Z., Trappe, W., Zhang, Y., and Nath, B., "Robust statistical methods for securing wireless localization in sensor networks". In Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05), 2005.

[8] Liu D., Ning P., Du Kevin, "Attack-Resistant Location Estimation in Sensor Networks". IEEE Fourth International Symposium on Information Processing in Sensor Networks, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1440904, 2005.

[9] Liu D., Ning P., Wang C., Du Kevin, "Attack-Resistant Location Estimation in Wireless Sensor Networks". ACM Transactions on Information and Systems Security, Vol. 11, No. 4, Article 22, http://doi.acm.org/10.1145/1380564.1380570, July 2008.

[10] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices". In IEEE Personal Communications Magazine, 7(5):28-34, October 2000.

[11] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System". In Proceedings of the IEEE INFOCOM ’00, March 2000.

[12] R. Nagpal, "Organizing a Global Coordinate System from Local Information on an Amorphous Computer". A.I. Memo1666, MIT A.I. Laboratory, August 1999.

[13] Ray, S., Ungrangsi, R., Pellegrini, F. D., Trachtenberg, A., and Starobinski, D. 2003. "Robust location detection in emergency sensor networks". In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM’03), 2003.

[14] Sastry, N., Shankar, U., and Wagner, D. 2003. "Secure verification of location claims". In Proceedings of the ACM Workshop on Wireless Security (WiSe’03).

[15] Scott, Jim, “Point In Triangle Test”. www.BlackPawn.com, http://www.blackpawn.com/texts/pointinpoly/default.html.

[16] Sun SPOT development team, “Frequently Asked Questions”. Sun SPOT World, Sun Microsystems, Inc. http://www.sunspotworld.org/docs/general-faq.html

[17] Sun SPOT development team, “Solarium User’s Guide, Red Release 5.0”. Sun Microsystems, Inc. Sun Labs July 2009. http://www.sunspotworld.org/docs/Red/SolariumUsersGuide.pdf

[18] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-free localization schemes in large scale sensor networks. In Proceedings of ACM MobiCom ’03, 2003.

[19] Weisstein, Eric W., "Barycentric Coordinates." From MathWorld, a Wolfram Web Resource. http://mathworld.wolfram.com/BarycentricCoordinates.html